Current amp provides frequency modulation

M PEREIRA AND A DEL RIO, UNIVERSITY OF VIGO, SPAIN

You can use current-feedback op amps in many VHF-band RF applications. Thanks to their 150-MHz bandwidth, current-feedback op amps can simplify your design task by replacing discrete components, such as bipolar transistors and FETs, in high-frequency circuits. The circuit in **Figure 1** is a frequency modulator that generates a 55-MHz VHF signal. The circuit is basically a VCO that displays little amplitude modulation over a wide frequency range. The maximum amplitude variation over the range of 50 to 60 MHz is less than 0.3 dB. Also, the circuit's frequency-vs-voltage transfer function has two linear zones: approximately 0.7 MHz/V in the lower range and 0.4 MHz/V in the upper range. **Table 1** shows the frequency-vs-voltage relationship. The design shows low sensitivity to power-supply variations or supply noise. The transfer-function changes are negligible for supply voltages from ±8 to ±12V.

TABLE 1—CARRIER FREQUENCY VS MODULATION VOLTAGE

Voltage (V)	Frequency (MHz)
–10	49.8
–8	51.2
–6	52.6
–4	53.8
–2	55
0	56.2
2	57
4	58
6	58.8
8	59.4

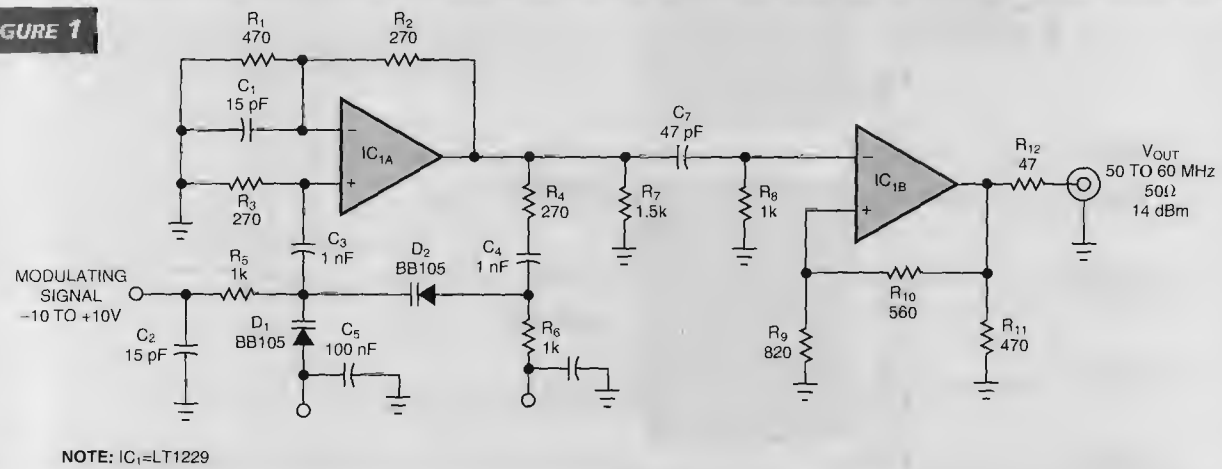
The design is compact, comprising an eight-pin IC, a few resistors and capacitors, and no inductors. The circuit comprises two blocks. The first block is the VCO, which uses two BB105 varicaps (D_1 and D_2) with IC_{1A}. The control voltage to these diodes can vary from 2 to 20V. You can change the frequency range by varying R_3 and R_4. The output frequency increases when those resistors decrease in value. If the value of R_{10} in the second-block gain stage decreases, the amplitude of the FM signal also decreases. The input modulating signal has a limited bandwidth because of the capacitors connected to the varicaps. The 1-nF capacitors in **Figure 1** exhibit a low impedance to the carrier and a high impedance to the input signal.

With the values shown, the circuit can accommodate input signals with a 30-kHz bandwidth. You can thus modulate two or three audio channels. The FM signal has a bandwidth that varies with the amplitude and bandwidth of the input. The second harmonic in the output signal is lower than –30 dB; the third harmonic, approximately –30 dB. The third harmonic is somewhat higher than the second because of clipping from the high gain. You can reduce the harmonics by connecting a lowpass filter to the output.

The second circuit block isolates the output from the VCO. A low or reactive impedance connected directly to the VCO could prevent oscillation or cause the VCO to oscillate at undesired frequencies. The IC_{1B} buffer provides the isolation. The buffer provides 14 dBm of gain to the FM signal, with an output stage adapted for a load of 50Ω. You could also use this circuit in frequency synthesizers, which comprise a PLL, phase comparators, and a VCO. Such synthesizers are especially useful in digital tuners. (DI #2122) EDN

To Vote For This Design, Circle No. 406

FIGURE 1



A current-feedback amplifier and a couple of varicaps make up a simple frequency modulator for multichannel-audio applications.

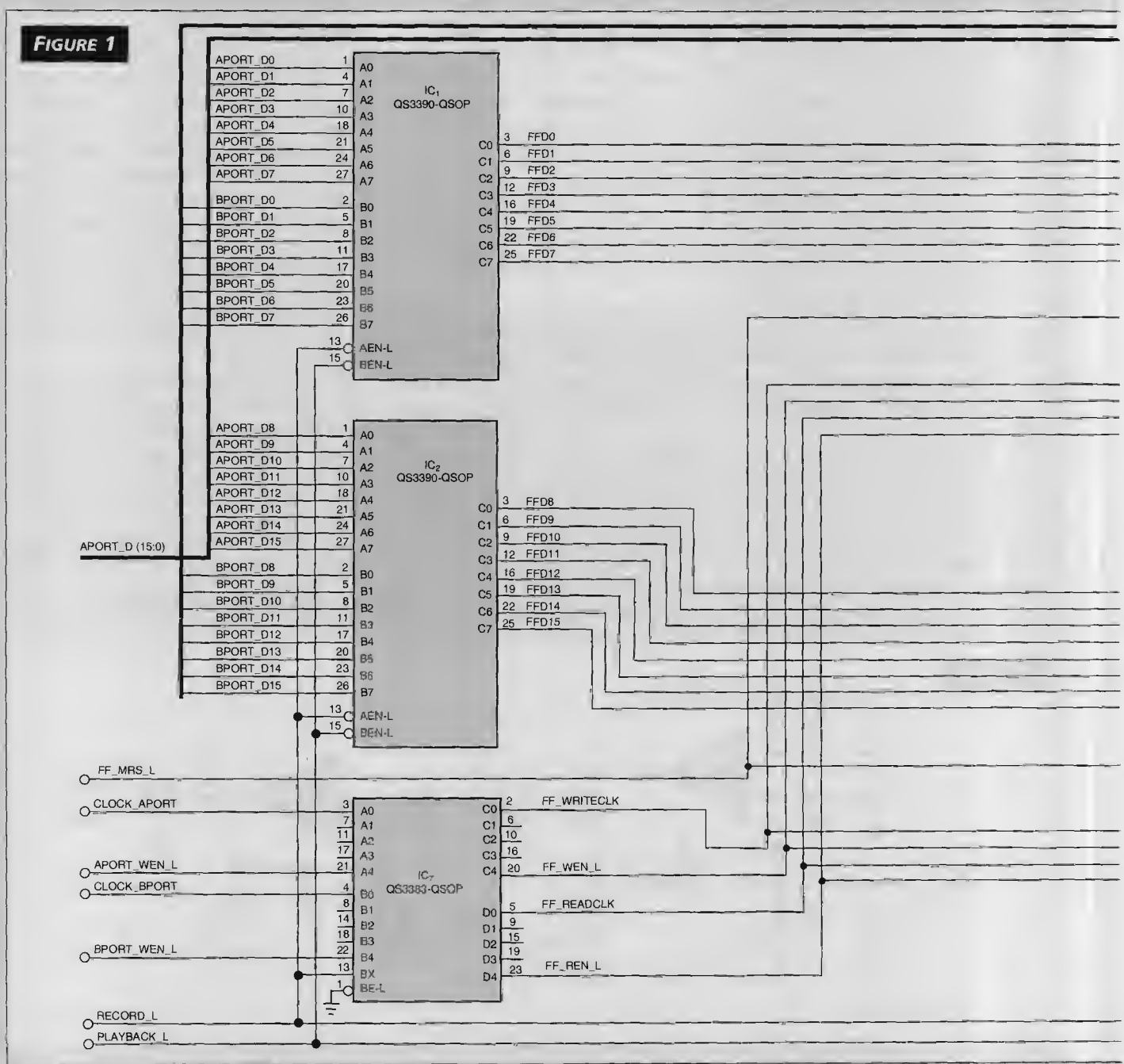
Multiplexers convert deep FIFO buffer to bidirectional

KEVIN KELLEY, TELEPHONICS CORP, FARMINGDALE, NY

Interfaces to digital data recorders often require deep FIFO buffers to match the system's data rate to the recorder rate. Sometimes, a second set of FIFO buffers in the opposite direction handles playback data. Doubling the FIFO-buffer size in this manner has obvious disadvantages in cost, power consumption, and board area. Inasmuch as you need only one data-flow direction at a time, you can use one FIFO buffer (**Figure 1**). The circuit converts a 32k×16 SuperSync FIFO design to bidirectional (half-duplex) using QuickSwitch QS3390 16-to-8 multiplexer/demultiplexer ICs. This implementation has the following advantages over the traditional tristate-multiplex/demultiplex approach:

- Near-zero propagation delays through the multiplexers produce minimal impact on high-speed data-flow tim-

FIGURE 1



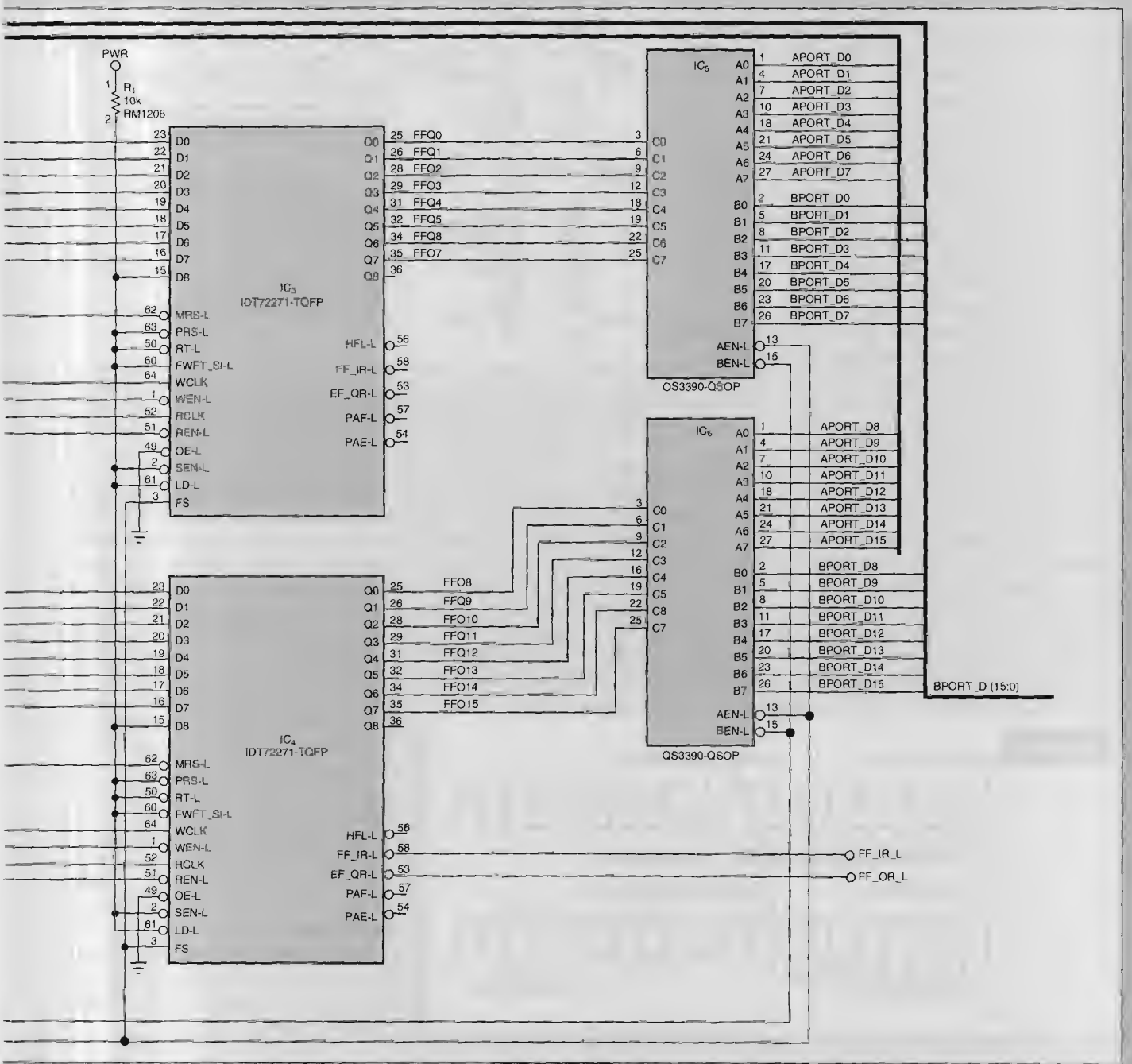
Using 16-to-8 multiplexer/demultiplexer ICs allows you to use one FIFO bank instead of two in playback/record systems.

ing through the FIFO buffer in either direction;

- This implementation requires fewer ICs: four QS3390s vs eight octal tristate buffer ICs (note that QS3390s are 28-pin devices and octal buffers are 20-pin ICs); and
- It consumes less power than a design using conventional buffers, because activity on the A, B, and C pins of the multiplexers does not contribute to the power dissipation of the devices, which consume 9 mA maximum each.

In this circuit, the Record_L and Playback_L signals are mutually exclusive; at any time, a low state determines the direction of data. When Record_L is low, it selects the A port to write into the FIFO buffer along with the A-port clock; the B port receives the FIFO data along with the B-port clock. When Playback_L is low (Record_L is high), the B port writes to the FIFO buffer and the A port receives the data. The QS3383 switches the FIFO read and write clocks and enables the FIFO buffer to match the ports. To avoid bus contention, you should tristate the devices driving the A and B ports during direction switching. (DI #2105) EDN

To Vote For This Design, Circle No. 407



Software speeds 8051 RS-232C receiver

JERZY CHRZASZCZ, WARSAW UNIVERSITY OF TECHNOLOGY, POLAND

This method allows for receiving data from a half-duplex, 9600-bps or faster serial link in an 8051-based system without using a hardware UART. The method's goal is to minimize performance degradation that software-service routines incur. Bit-cell time for 9600-bps transmission with 8 data bits, no parity, and 1 stop bit (8N1 format) is approximately 104 µsec. This rate translates into 96 processor cycles if you use an 11.0592-MHz crystal. Thus, 960 processor cycles are available during byte-frame reception.

The proposed solution is based on **Reference 1**. The technique uses external interrupt INT1 as a data-receive line. Timer T0 generates the sampling timebase using mode 2 (8-bit autoreload) (**Figure 1**). In idle state, the timer remains stopped, and the external interrupt enables. The start of an incoming data frame produces an INT1 request. The interrupt-service routine loads, starts the timer, and disables INT1. Upon initiation of the process, the first timer interrupt verifies the start bit, so the initial delay is approximately half the bit-cell time. Then, the system samples 8 data bits and, during the service of the tenth timer interrupt, validates the stop bit. The timer stops, and INT1 re-enables to ready the system for the next byte (**Figure 1a**).

A receive sequence consumes 264 processor cycles. The time between "data ready" and clearing the received byte after identifying the next start bit is only approximately 90 cycles for back-to-back transmission. If the main program cannot accept data within that time, an overrun error arises. The modified code in **Listing 1** is the receive-related part of an 80C51 service routine that trades framing-error detection for execution speed. You can download this listing, as well as other associated files, from *EDN*'s Web site, www.ednmag.com. For example, an assembly-code file gives a detailed transmit/receive example. At the registered-user area, go into the "Software Center" to download the files from DI-SIG, #2125.

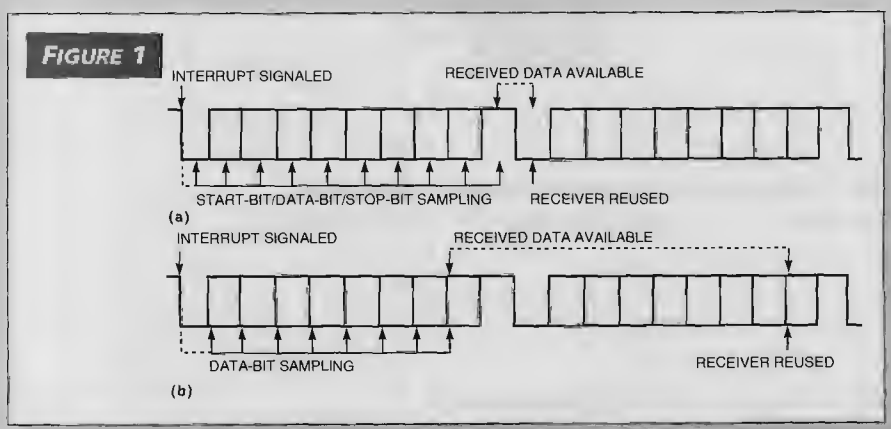
LISTING 1—MODIFIED 8051 CODE FOR SPEED ENHANCEMENT IN UART RECEIVER

```
; timer 0 timeout - rsload rsgister TH0 is initializsd to bit cell time
Tr0Int:         pueh    ACC
        pueh    PSW
        jb      RxBuey,RxBit    ; ie thie a recsive timer interrupt?
        ;.......                ; anything sles?
        pop     PSW
        pop     ACC
        rsti

; begin eoft rsceivs (after sxternal interrupt 1)
Ex1Int:         jb      Rx,Ex1Ret       ; oops, this was not a start bit
        mov     TL0,#STRTVAL    ; this ie a start bit - load first timeout
        estb    TR0             ; etart timer
        mov     BitCnt,#08      ; sst bit count (WITHOUT START & STOP)
        clr     EX1             ; dieable sxtsrnal interrupt 1
        sstb    RxBusy          ; est rsceive-in-progrees flag
Ex1Ret:         reti

; eoft rsceive bit routins
RxBit:          mov     A,RxShift       ; gst partial receive byte
        mov     C,Rx            ; get recsive pin value
        rrc     A               ; ehift in new bit
        mov     RxShift,A       ; save updatsd receivs byte
        djnz    BitCnt,RxRst    ; decrement bit count, test for stop
        mov     RxData,RxShift  ; eavs recsivsd data
        clr     TR0             ; stop timer
        clr     RxBuey          ; rslease timer for other purposee
        clr     IE1             ; clear INT1 edge dstect flag
        setb    EX1             ; enable sxternal interrupt 1
        sstb    RxFull          ; tell mainline that a byte ie ready
RxRst:          pop     PSW
        pop     ACC
        rsti
```

The technique does not verify start and stop bits and samples only data bits on timer ticks. The external interrupt is edge-sensitive to avoid false triggering when the last data bit is 0. Consequently, the bit-receive procedure is simple and fast. The data-ready condition becomes flagged after reception of the last data bit, and, because the sampling point resides at the beginning of the bit cell, the flagging occurs approximately 1.5 bit cells earlier than it does in the original code (**Figure 1b**). Such timing increases the chance to use the data, but just to be sure, the method copies the byte to another memory location. This technique mimics double-buffer operation of a hardware UART and gives the processor an additional 1.04 msec when it receives the next byte. The reworked receive sequence lasts 179 cycles, representing an approximate 30% speed improvement. (DI #2125)



A modification of an 8051's input-sampling method (b) results in an approximate 30% speed improvement over the original method (a).

Reference

Application Note AN423, "Software driven serial communication routines for the 83C751 and 83C752," Philips Semiconductors.

To Vote For This Design, Circle No. 408

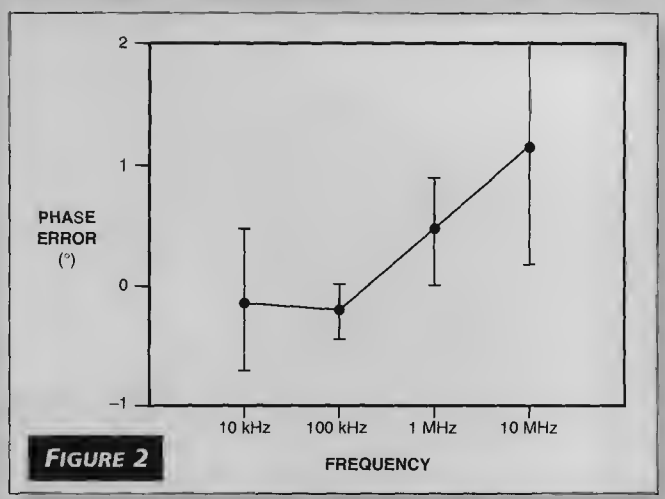
DDS forms inphase and quadrature generator

J TODA, R BRAGOS, AND M TRESANCHEZ, UPC, BARCELONA, SPAIN

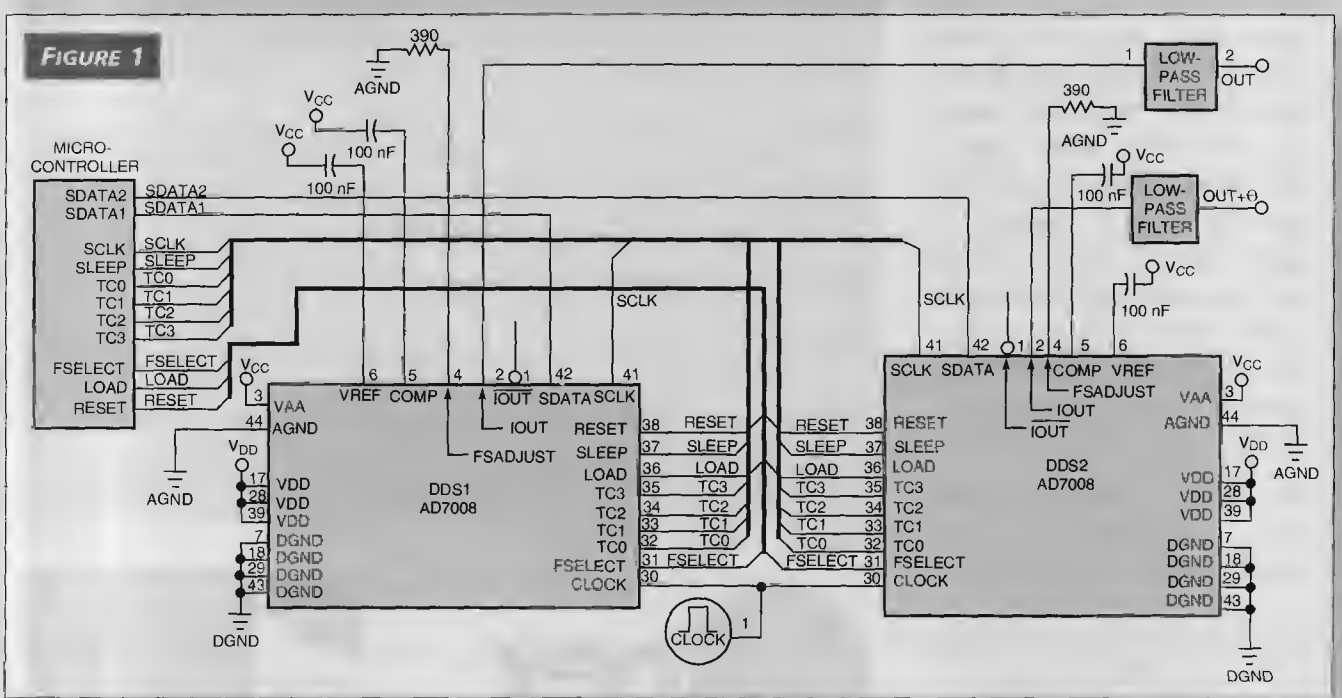
Many instrumentation applications use two sine waves to carry out coherent modulation. The circuit in **Figure 1** allows you to measure vector magnitudes (for example, voltage or impedance) by multiplying the measured signal with inphase and quadrature sinusoidal signals of the same frequency. In a classic approach, these signals come from one VCO and a $\pi/2$ (90°) delay. However, when the measurements cover a wide frequency span, it is necessary to compensate phase differences between the measured and instrument circuits by using a programmable delay. The problem is that a programmable-delay circuit usable across large frequency spans (for example, seven decades) is difficult to implement.

This circuit solves the problem by using two chips. The circuit takes little board space and provides precise phase matching. Direct digital synthesis (DDS) allows you to generate periodic signals using a numerically controlled oscillator (NCO) and a DAC. The NCOs' look-up table stores a set of sinusoidal values. Reading these values in appropriate order, the system generates a sinusoidal signal. A phase register changes the table-reading speed and produces different frequencies without affecting the sampling rate. You can also program the DAC to produce different amplitudes.

A DDS system is suitable for generating sine waves over large frequency spans, with programmable phase delay and amplitude if desired. Also, some DDS chips (for example, the AD7008) include the DAC output. This inclusion is convenient in instrumentation applications, because it reduces the number of noisy, high-frequency digital lines. However, this IC has two analog outputs with a fixed 180° phase shift between them. Thus, to generate the quadrature signals, you need two DDS systems. However, a new problem arises: the lack of a synchronization mechanism for a set of AD7008s. The chip's sleep line stops the NCO's internal clock, but this



The circuit in Figure 1 yields phase errors of less than 1° over most of its operating-frequency range.



Two DDS chips allow you to generate inphase and quadrature sinusoidal signals over a wide frequency range with low phase error.

utility is inadequate to synchronize the phases of the two DDS systems.

The circuit of **Figure 1** uses a passive synchronization system based on simultaneous data load. The circuit uses the serial port of the DDS chip; you could also use the faster parallel port. The method shares all the lines used for programming the DDS ICs except the data lines. You can thus simultaneously program the two AD7008s with different data; that is, with a different phase delay for each chip. The only restriction is that you must route the critical timing lines (reset, clock, and load) in such a way that the line lengths from the signal source (the 50Ω clock) to the signal sink (the two chips' clock inputs) are equal.

Also, note that the μC's timing signal (reset) must be synchronous with the clock signal of the DDS chips. **Figure 2** shows the phase error between the inphase and quadrature signals. The error is less than 0.1° for frequencies as high as 100 kHz, less than 0.5° to 1 MHz, and less than 1.5° at 10 MHz. The vertical lines show the standard deviation for each measurement. (DI #2126)

To Vote For This Design, Circle No. 409

RS-232C port scans remote keypad

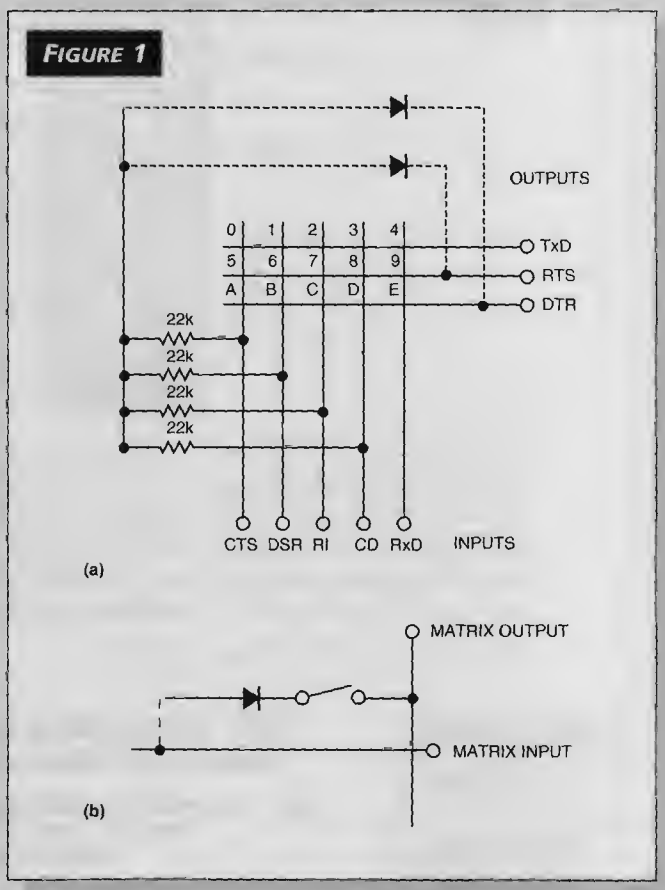
SK SHENOY, NPOL, KOCHI, INDIA

In many embedded or control applications, you might prefer to use a small, customized keypad with your PC-based system rather than a full-fledged alphanumeric keyboard. You may also need to locate the keypad some distance from the system hardware. If you have no parallel ports available for this purpose, you could use a standard RS-232C port. With the method shown here, you need only some wire and a few passive components to connect a keypad with as many as 15 keys to the RS-232C port, at distances to 50 ft. **Figure 1a** shows the basic connection; **Figure 1b** shows the connection of the keypad switches in the matrix.

A full RS-232C port has two modem-control output lines (RTS and DTR) and four modem-status lines (CTS, DSR, CD, and RI), in addition to the RxD and TxD data lines. You can make the DTR and RTS lines active or inactive under software control by writing to the modem-control register of the UART. However, many people are unaware that you can similarly control the TxD line if the UART supports the break-generation utility. Similarly, you can sample the modem-status input line at any time by reading the UART's modem-status register. Thus, you can use these RS-232C input and output lines as a key-scanning matrix.

According to the RS-232C standard, a receiver reads any input above 3V as active and below –3V as inactive. To avoid transients from floating inputs, ORing diodes (with broken lines in **Figure 1**) pull the inputs to the inactive (–12V) state through 22-kΩ resistors connected to the RTS and DTR modem-control lines (at least one of which is inactive while scanning). The pulldown resistors provide added insurance; in most cases, the circuit works perfectly without them. You can connect diodes at the key matrix crosspoints for short-circuit protection if you simultaneously press an active key and an inactive key. If your RS-232C drivers have built-in short-circuit protection, as most do, these diodes are unnecessary.

A Borland Version 2.0 C program illustrates the scanning of a 3×5 keypad connected to a PC's serial port based on an 8250 UART. You can download the listing from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the "Software Center" to download the files from DI-SIG, #2128. The idea works with any PC system and UART with similar I/O lines. The lines you need are the TxD, DTR, RTS, DSR, RI, CTS, and RxD. The first three lines are outputs, and they activate cyclically (12V) one at a time. The active peri-



Some wire, a few passive components, and an RS-232C port (a) are all you need to replace a full-featured keyboard with a small keypad. The keypad switches insert a diode at the associated matrix crosspoint (b).

od is typically a few milliseconds, but this parameter depends on how fast you want to scan, the debouncing aspects, and other considerations.

Depending on what key you press, one of the input pins becomes active; the UART's modem-status register (for CTS, DSR, CD, and RI) reads it. Unlike the modem-status lines, however, the UART cannot sample the RxD line. However, depending on how long you press the key and on the overlap of the key-press time with the transmission of the Null character (12V on DTR/RTS), either a full Null character or only a fragment transfers to the RxD line. In any case, the Rx data-ready flag sets if the duration of the space (12V) on the RxD line exceeds the start-bit time. Only the data read (which is immaterial) varies, depending on the aforementioned factors.

If the duration of the space on the receive line is longer than a character time, including start and stop bits, the break-detected status line also sets. Note that, in the worst case, the Rx data-ready flag sets eight times after the output line deactivates. Therefore, to avoid ambiguity, the activation of the next output line occurs only after waiting for this delay and then checking the Rx data-ready flag. You can effect key debouncing by verifying that the same key is active over a few repeated scans before signaling the press as active. If the UART does not support the break-generation utility, you need only transmit a Null (00 hex) character with the baud rate set to obtain a space of the required duration on the TxD line. (DI #2128) EDN

To Vote For This Design, Circle No. 410

±12V supply accepts 9 to 30V inputs

RON YOUNG, MAXIM INTEGRATED PRODUCTS, SUNNYVALE, CA

For applications that require symmetric supply voltages from a wide input range, the switch-mode supply in **Figure 1** delivers more than 3.6W at ±12V and operates from inputs of 9 to 30V. Over this input range, efficiency varies from 74 to 80%. The step-down switching regulator, IC_1, includes an internal switching transistor that drives the primary side of the flyback transformer. Matched output windings produce the ±12V outputs. Resistor-divider feedback by R_2 and R_3 regulates the positive output, and tight coupling between the output windings regulates the negative output.

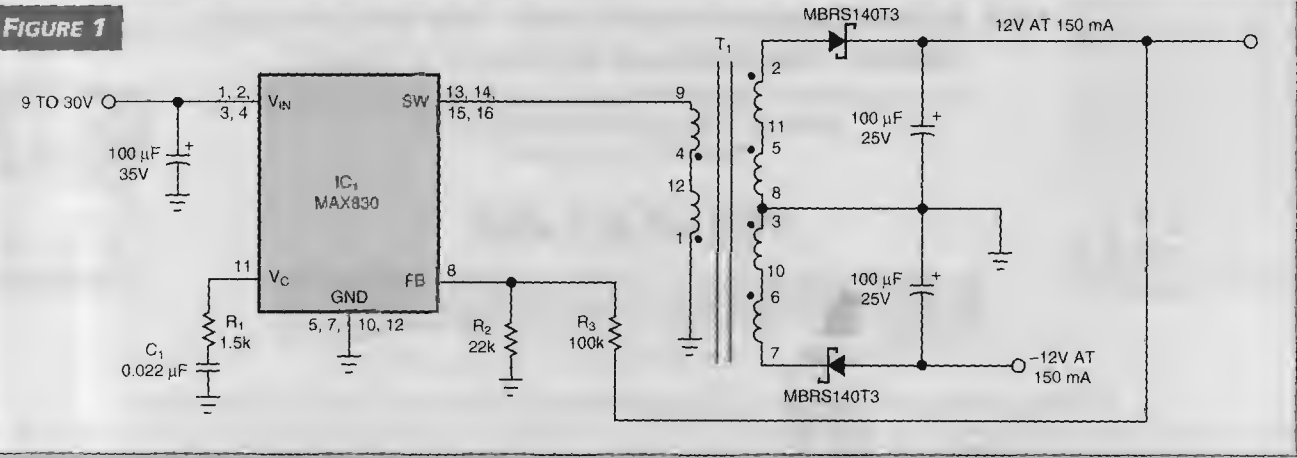
With a 20-mA load on the positive 12V output, the –12V output changes less than 7% in response to a load change of 0 to 300 mA. In other words, cross-regulation is less than 7%. You can adjust the feedback resistors for different output voltages and optimize the R_1/C_1 compensation network for your application and layout.

The close-coupled transformer windings provide another benefit: The primary winding's low leakage inductance requires no snubber circuitry for control of flyback voltage to protect the internal transistor switch. The switch voltage flies back to the reflected output plus a diode drop, or to a total of –13V, which is well within the IC's absolute maximum voltage rating.

The primary winding comprises two windings in series to achieve its nominal 44-µH inductance. To maintain a 1-to-1-to-1 turns ratio, each of the output windings comprises two windings in series. (DI #2121) EDN

To Vote For This Design, Circle No. 411

FIGURE 1



A step-down switching regulator and matched output windings produce ±12V outputs from a 9 to 30V input.

Conversion program outputs BCD values

MIKE MCGLINCHY, MDM DESIGNS, SAN BRUNO, CA

A program entitled HEXBCD is an 8051 µC subroutine that converts a 16-bit HEX number in internal RAM locations 33 and 34H into its BCD equivalent (**Listing 1**). The program places the result in scratchpad registers R1, R2, and R3 . (You can download the program from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2120.)

Many applications require a µC to output BCD values, such as for driving a BCD seven-segment LED decoder. Converting an 8-bit number to BCD is relatively simple; the routine LOW8, which is part of the main HEXBCD program, performs an 8-bit conversion. You can add more registers to expand the overall conversion program beyond 16 bits to handle larger numbers.

The maximum time to perform a 16-bit conversion is 236 instruction cycles. With a 12-MHz crystal and the corre-

LISTING 1—HEXBCD CONVERSION PROGRAM

```
HEXBCD:
        CLR A                   ;zero BCD result
        MOV R1,A                ;registers R1, R2 and R3.
        MOV R2,A
        MOV R3,A
        MOV A,33H               ;Get HIGH byte of hex #.
        JNB ACC.7,BIT6          ;If bit 7 of HIGH byte
                                ;is NOT high, test next bit.
        MOV R1,#003H            ;It's HIGH, so put in 32768H.
        MOV R2,#027H
        MOV R3,#068H
;-------------------------------------------------------
BIT6:   JNB ACC.6,BIT5          ;If bit 6 of HIGH byte is
                                ;NOT high, test bit 5.
        MOV R4,#001H            ;It's HIGH, so
        MOV R5,#063H
        MOV R6,#084H
        CALL BCDADD             ;Add 16384H to R1, R2 and R3.
;-------------------------------------------------------
BIT5:   JNB ACC.5,BIT4          ;If bit 5 of HIGH byte is
                                ;NOT high, test bit 4.
        MOV R5,#081H            ;It's HIGH, so add 8192H to
        MOV R6,#092H            ;R1,R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
BIT4:   JNB ACC.4,BIT3          ;If bit 4 of HIGH byte is
                                ;NOT high, test bit 3.
        MOV R5,#040H            ;It's HIGH, so add 4096H
        MOV R6,#096H            ;to R1,R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
BIT3:   JNB ACC.3,BIT2          ;If bit 3 of HIGH byte is
                                ;NOT high, test bit 2.
        MOV R5,#020H            ;It's HIGH, so add 2048H
        MOV R6,#048H            ;to R1, R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
BIT2:   JNB ACC.2,BIT1          ;If bit 2 of HIGH byte is
                                ;NOT high, test bit 1.
        MOV R5,#010H            ;It's HIGH, so add 1024H
        MOV R6,#024H            ;to R1, R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
BIT1:   JNB ACC.1,BIT0          ;If bit 1 of HIGH byte is
                                ;NOT high, test bit 0.
        MOV R5,#005H            ;It's HIGH, so add 512H
        MOV R6,#012H            ;to R1, R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
BIT0:   JNB ACC.0,LOW8          ;If bit 0 of HIGH byte is
                                ;NOT high, test lower byte.
        MOV R5,#002H            ;It's HIGH, so add 256H
        MOV R6,#056H            ;to R1, R2 and R3.
        CALL BCDADD
;-------------------------------------------------------
;       LOW8-Does the lower 8 bits hex-bcd conversion
;-------------------------------------------------------
LOW8:   MOV A,34H               ;Get low byte of hex # from
                                ;loc 34H.
        MOV B,#100              ;find out how many 100's.
        DIV AB
        MOV R5,A                ;R5 holds BCD hundreds
        MOV A,#10
        XCH A,B                 ;find out # of 10's.
        DIV A
        SWAP A
        ADD A,B
        MOV R6,A                ;R6 holds BCD 10's and
                                ;1's digits.
        CALL BCDADD             ;add lower byte BCD # to
                                ;upper byte results
                                ;answer in R1, R2 and R3.
DONE:   RET                     ;16 bit conversion complete!

;*******************************************************
;       BCDADD-adds R4,R5 and R6 to R1, R2 and R3
;*******************************************************
BCDADD:
        PUSH ACC                ;Save Acc.
        MOV A,R6                ;Get low BCD digits
        ADDC A,R3
        DA A
        MOV R3,A                ;low digits BCD answer in R3

        MOV A,R5                ;Get middle BCD digits
        ADDC A,R2
        DA A
        MOV R2,A                ;middle digits BCD answer in R2

        MOV A,R4                ;Get high BCD digit
        ADDC A,R1
        DA A
        MOV R1,A                ;high digits BCD answer in R1

        MOV R4,#000H            ;Zero R4 in case it's not
                                ;used again.
        CLR C                   ;Clear Carry flag
        POP ACC                 ;restore Acc.
        RET
;*************************************************************

        END
```

sponding 1 μsec/instruction, the maximum time is then 236 μsec. The more 1s that exist in the number, the longer the conversion takes. For example, converting FFFFH (all 1s) to its BCD equivalent (65,535) takes the maximum of 236 μsec. Converting 4104H to BCD takes 113 μsec because this HEX number contains only three 1s.

The program begins by clearing the BCD registers R1, R2, and R3. The program then takes the high byte of the 16-bit HEX number from RAM location 33H and tests each bit for a high. If a bit is high, the program places a corresponding binary-weighted number in temporary-storage registers R4, R5, and R6. For example, if bit 6 (bit 14 overall) of the high byte is high, the program places 16384H in R4, R5, and R6. The program then calls a subroutine, BCDADD, which adds 16384H to the contents of R1, R2, and R3 using the ADDC A,R1 (add register to accumulator with carry) and DA A (decimal adjust accumulator) instructions.

After the program tests and adds the upper 8 bits, the intermediate BCD answer is in locations R1, R2, and R3. The routine LOW8 then does a quick HEX-to-BCD conversion of the lower 8 bits. The program adds this result to the high-byte result by calling BCDADD once more. (DI #2120) EDN

To Vote For This Design, Circle No. 412

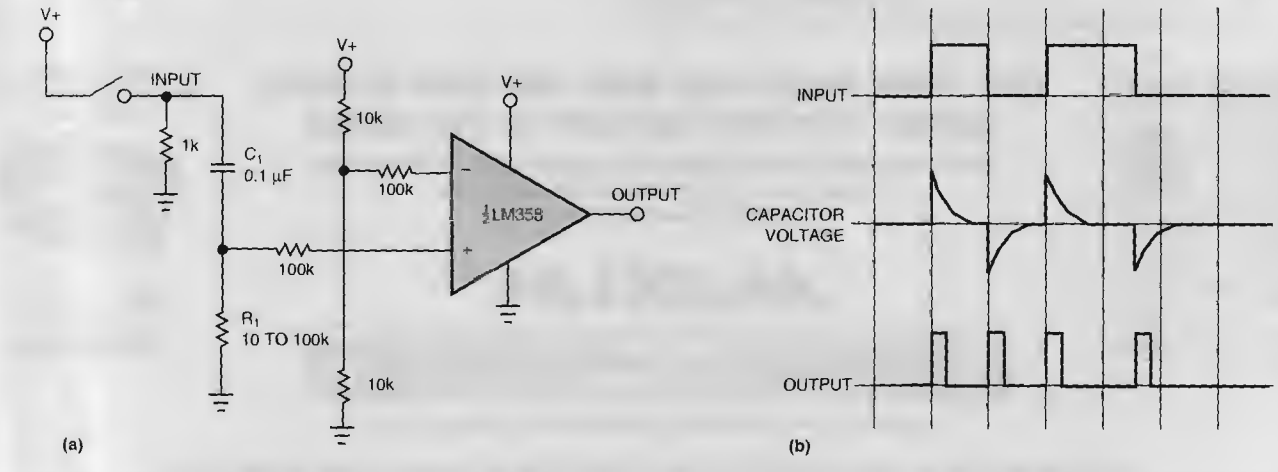
Edge detector runs off of single supply

JOE MELOCHE, INGERSOLL-RAND FLUID PRODUCTS DIVISION, BRYAN, OH

You can use a single-supply circuit to generate a pulse on both the rising and the falling edges of a signal for use with counters or similar devices that require only a rising edge to trigger (**Figure 1a**). R_1 and C_1 form a differentiator that converts rising edges to positive spikes and falling edges to negative spikes. This output drives the LM358, which the circuit configures as a comparator with a reference input equal to ½V+. As expected, the LM358 converts the positive spike to a positive square pulse of approximately 2.5τ in duration, where $\tau=R_1\times C_1$. The LM358 also converts the negative spikes to similar positive pulses because of the LM358's characteristic ability to operate as a fully functional single-supply op amp (**Figure 1b**). Tests using LM358s from National Semiconductor (Santa Clara, CA) and Motorola (Austin, TX) confirm proper operation. (DI #2116) EDN

To Vote For This Design, Circle No. 413

FIGURE 1



A single-supply op amp (a) operating as a comparator teams with differentiator R_1/C_1 to produce positive pulses at both rising and falling edges of the input (b).